

Computational Paths and Identity Types

Arthur F. Ramos* Ruy J. G. B. de Queiroz*
Anjolina G. de Oliveira*

* Centro de Informática
Universidade Federal de Pernambuco
afr@cin.ufpe.br
ruy@cin.ufpe.br
ago@cin.ufpe.br

Abstract

We introduce a new way of formalizing the intensional identity type based on the notion of computational paths which will be taken to be proofs of propositional equality, and thus terms of the identity type. Our approach results in an elimination rule different than the one given by Martin-Löf in his intensional identity type. In order to show the validity and power of our approach, we formulate and prove the basic concepts, lemmas and theorems of Homotopy Type Theory using computational paths. We also show that these proves and formulations resulted, as a side effect, in the improvement, by means of the addition of new rules, of a term rewrite system known as $LND_{EQ} - TRS$, originally proposed by de Oliveira (1995).

Keywords. Identity type, computational paths, equality proofs, path-based constructions, type theory, term rewriting systems, homotopy type theory.

1 Introduction

One interesting peculiarity of Martin-Löf's Intensional Type Theory is the existence of two distinct kinds of equalities between terms of the same type. The first one is originated by the fact that equality can be seen as a type. The second one is originated by the fact that two terms can be equal by definition.

The treatment of an equality as a type gives rise to an extremely interesting type known as identity type. The idea is that, given terms a, b of a type A , one may form the type whose elements are proofs that a and b are equal elements of type A . This type is represented by $Id_A(a, b)$. A term $p : Id_A(a, b)$ makes up for the *grounds* (Prawitz, 2009) (or proof) that establishes that a is indeed equal to b . We say that a is *propositionally* equal to b .

The second kind of equality is called definitional and is denoted by \equiv . It occurs when two terms are equal by definition, i.e., there is no need for an evidence or a proof to establish the equality. A classic example, given in Univalent Foundations Program (2013), is to consider any function definition, for example, $f : \mathbb{N} \rightarrow \mathbb{N}$ defined by $f(x) \equiv x^2$. For $x = 3$, we have, by definition, that $f(x) \equiv 3^2$. We are unable to conclude that $f(3) \equiv 9$ though. The problem is that, to conclude that $3^2 = 9$, we need additional evidence. We need a way to compute 3^2 , i.e., we need to use exponentials (or multiplications) to conclude that 3^2 is equal to 9. That way, we can only prove that 3^2 is propositionally equal to 9, i.e., there is a $p : Id_{\mathbb{N}}(3^2, 9)$.

Between those two kind of equalities, the propositional one is, without a doubt, the most interesting one. This claim is based on the fact that many interesting results have been achieved using the identity type. One of these was the

discovery of the Univalent Models in 2005 by Vladimir Voevodsky (Voevodsky, n.d.). A groundbreaking result has arisen from Voevodsky's work: the connection between type theory and homotopy theory. The intuitive connection is simple: a term $a : A$ can be considered as a point of the space A and $p : Id_A(a, b)$ is a homotopy path between points $a, b \in A$ (Univalent Foundations Program, 2013). This has given rise to a whole new area of research, known as Homotopy Type Theory. It leads to a new perspective on the study of equality, as expressed by Voevodsky in a recent talk in *The Paul Bernays Lectures* (Sept 2014, Zürich): equality (for abstract sets) should be looked at as a *structure* rather than as a *relation*.

Motivated by the fact that the identity type has given rise to such interesting concepts, we have been engaged in a process of revisiting the construction of the intensional identity type, as originally proposed by Martin-Löf. Although beautifully defined, we have noticed that proofs that uses the identity type can be sometimes a little too complex. The elimination rule of the intensional identity type encapsulates lots of information, sometimes making too troublesome the process of finding the reason that builds the correct type.

Inspired by the path-based approach of the homotopy interpretation, we believe that a similar approach can be used to define the identity type in type theory. To achieve that, we have been using a notion of *computational paths*. The interpretation will be similar to the homotopy one: a term $p : Id_A(a, b)$ will be a computational path between terms $a, b : A$, and such path will be the result of a sequence of rewrites. In the sequel, we shall define formally the concept of a computational path. The main idea, i.e. proofs of equality statements as (reversible) sequences of rewrites, is not new, as it goes back to a paper entitled "Equality in labeled deductive systems and the functional interpretation of propositional equality", presented in December 1993 at the *9th Amsterdam Colloquium*, and published in the proceedings in 1994 (de Queiroz & Gabbay, 1994).

After establishing the connection between the identity type and computational paths, we investigate well established properties and concepts of the foundations of homotopy type theory. We are interested in the ones connected to the identity type. Our main objective is to show that these properties and theorems are valid in our path-based approach for the identity type. In this sense, our objective is to show that one can use computational paths to define and develop concepts of homotopy type theory.

2 Computational Paths

Since computational path is a generic term, it is important to emphasize the fact that we are using the term computational path in the sense defined by de Queiroz & de Oliveira (2014). A computational path is based on the idea that it is possible to formally define when two computational objects $a, b : A$ are equal. These two objects are equal if one can reach b from a applying a sequence of axioms or rules. This sequence of operations forms a path. Since it is between two computational objects, it is said that this path is a computational one. Also, an application of an axiom or a rule transforms (or rewrite) an term in another. For that reason, a computational path is also known as a sequence of rewrites. Nevertheless, before we define formally a computational path, we can take a look at one famous equality theory, the $\lambda\beta\eta$ - equality (Hindley & Seldin, 2008):

Definition 2.1. The $\lambda\beta\eta$ -equality is composed by the following axioms:

(α) $\lambda x.M = \lambda y.[y/x]M$ if $y \notin FV(M)$;

(β) $(\lambda x.M)N = [N/x]M$;

(ρ) $M = M$;

(η) $(\lambda x.Mx) = M$ ($x \notin FV(M)$).

And the following rules of inference:

(μ) $\frac{M = M'}{NM = NM'}$ (τ) $\frac{M = N \quad N = P}{M = P}$

(ν) $\frac{M = M'}{MN = M'N}$ (σ) $\frac{M = N}{N = M}$

(ξ) $\frac{M = M'}{\lambda x.M = \lambda x.M'}$

Definition 2.2. (Hindley & Seldin, 2008) P is β -equal or β -convertible to Q (notation $P =_\beta Q$) iff Q is obtained from P by a finite (perhaps empty) series of β -contractions and reversed β -contractions and changes of bound variables. That is, $P =_\beta Q$ iff **there exist** P_0, \dots, P_n ($n \geq 0$) such that $P_0 \equiv P$, $P_n \equiv Q$, $(\forall i \leq n-1)(P_i \triangleright_{1\beta} P_{i+1}$ or $P_{i+1} \triangleright_{1\beta} P_i$ or $P_i \equiv_\alpha P_{i+1})$.

(NB: equality with an **existential** force, which will show in the proof rules for the identity type.)

The same happens with $\lambda\beta\eta$ -equality:

Definition 2.3. ($\lambda\beta\eta$ -equality (Hindley & Seldin, 2008)) The equality-relation determined by the theory $\lambda\beta\eta$ is called $=_{\beta\eta}$; that is, we define

$$M =_{\beta\eta} N \Leftrightarrow \lambda\beta\eta \vdash M = N.$$

Example 2.4. Take the term $M \equiv (\lambda x.(\lambda y.yx)(\lambda w.zw))v$. Then, it is $\beta\eta$ -equal to $N \equiv zv$ because of the sequence:

$(\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z, zv$

which starts from M and ends with N , and each member of the sequence is obtained via 1-step β - or η -contraction of a previous term in the sequence. To take this sequence into a *path*, one has to apply transitivity twice, as we do in the example below.

Example 2.5. The term $M \equiv (\lambda x.(\lambda y.yx)(\lambda w.zw))v$ is $\beta\eta$ -equal to $N \equiv zv$ because of the sequence:

$(\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z, zv$

Now, taking this sequence into a path leads us to the following:

The first is equal to the second based on the grounds:

$\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v)$

The second is equal to the third based on the grounds:

$\beta((\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z)$

Now, the first is equal to the third based on the grounds:

$\tau(\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v), \beta((\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z))$

Now, the third is equal to the fourth one based on the grounds:

$\beta((\lambda y.yv)z, zv)$

Thus, the first one is equal to the fourth one based on the grounds:

$\tau(\tau(\eta((\lambda x.(\lambda y.yx)(\lambda w.zw))v, (\lambda x.(\lambda y.yx)z)v), \beta((\lambda x.(\lambda y.yx)z)v, (\lambda y.yv)z)), \beta((\lambda y.yv)z, zv)))$.

The aforementioned theory establishes the equality between two λ -terms. Since we are working with computational objects as terms of a type, we need to translate the $\lambda\beta\eta$ -equality to a suitable equality theory based on Martin L f's type theory. We obtain:

Definition 2.6. The equality theory of Martin L f's type theory has the following basic proof rules for the Π -type:

$$\begin{array}{ll}
(\beta) \quad \frac{[x : A] \quad N : A \quad M : B}{(\lambda x.M)N = M[N/x] : B[N/x]} & (\xi) \quad \frac{[x : A] \quad M = M' : B}{\lambda x.M = \lambda x.M' : (\Pi x : A)B} \\
(\rho) \quad \frac{M : A}{M = M : A} & (\mu) \quad \frac{M = M' : A \quad N : (\Pi x : A)B}{NM = NM' : B[M/x]} \\
(\sigma) \quad \frac{M = N : A}{N = M : A} & (\nu) \quad \frac{N : A \quad M = M' : (\Pi x : A)B}{MN = M'N : B[N/x]} \\
(\tau) \quad \frac{M = N : A \quad N = P : A}{M = P : A} & \\
(\eta) \quad \frac{M : (\Pi x : A)B}{(\lambda x.Mx) = M : (\Pi x : A)B} \quad (x \notin FV(M)) &
\end{array}$$

We are finally able to formally define computational paths:

Definition 2.7. Let a and b be elements of a type A . Then, a *computational path* s from a to b is a composition of rewrites (each rewrite is an application of the inference rules of the equality theory of type theory or is a change of bound variables). We denote that by $a =_s b$.

As we have seen in example 2.5, composition of rewrites are applications of the rule τ . Since change of bound variables is possible, consider that each term is considered up to α -equivalence.

3 Identity Type

In this section, we have two main objectives. The first one is to propose a formalization to the identity type using computational paths. The second objective is to show how can one use our approach to construct types representing reflexivity, transitivity and symmetry. In the case of the transitive type, we also compare our approach with the traditional one, i.e., Martin-L f's Intensional type. With this comparison, we hope to show the clear advantage of our approach, in terms of simplicity. Since our approach is based on computational paths, we will sometimes refer to our formulation as the *path-based* approach and the traditional formulation as the *pathless* approach. By this we mean that, even though the Homotopy Type Theory approach to the identity type brings about the notion of paths in the semantics, there is little in the way of handling paths as terms in the language of type theory.

Before the deductions that build the path-based identity type, we would like to make clear that we will use the following construction of the traditional approach (Harper, 2012):

$$\frac{A \text{ type} \quad a : A \quad b : A}{Id_A(a, b) \text{ type}} Id - F \quad \frac{a : A}{r(a) : Id_A(a, a)} Id - I$$

$$\frac{a : A \quad b : A \quad c : Id_A(a, b) \quad q(x) : C(x, x, r(x)) \quad [x : A] \quad [x : A, y : A, z : Id_A(x, y)] \quad C(x, y, z) \text{ type}}{J(p, q) : C(a, b, c)} Id - E$$

3.1 Path-based construction

The best way to define any formal entity of type theory is by a set of natural deductions rules. Thus, we define our path-based approach as the following set of rules:

- Formation and Introduction rules:

$$\frac{A \text{ type} \quad a : A \quad b : A}{Id_A(a, b) \text{ type}} Id - F \quad \frac{a =_s b : A}{s(a, b) : Id_A(a, b)} Id - I$$

- Elimination rule:

$$\frac{m : Id_A(a, b) \quad h(g) : C \quad [a =_g b : A]}{REWR(m, \acute{g}.h(g)) : C} Id - E$$

- Reduction rules:

$$\frac{\frac{a =_m b : A}{m(a, b) : Id_A(a, b)} Id - I \quad \frac{[a =_g b : A] \quad h(g) : C}{Id - E}}{REWR(m, \acute{g}.h(g)) : C} Id - E \quad \triangleright_\beta \quad \frac{[a =_m b : A] \quad h(m/g) : C}{h(m/g) : C}$$

$$\frac{e : Id_A(a, b) \quad \frac{[a =_t b : A] \quad t(a, b) : Id_A(a, b)}{Id - I} Id - E}{REWR(e, \acute{t}.t(a, b)) : Id_A(a, b)} Id - E \quad \triangleright_\eta \quad e : Id_A(a, b)$$

In these rules, \acute{g} (and \acute{t}) to indicate that they are abstractions over the variable g (or t), for which the main rules of conversion of λ -abstraction hold. For that reason, we proposed two reduction rules that handle these conversions, the β and η reduction rules.

Our introduction and elimination rules reassures the concept of equality as an **existential force**. In the introduction rule, we encapsulate the idea that an witness of a identity type $Id_A(a, b)$ only exists if there exist a computational path establishing the equality of a and b . Also, the elimination rule is similar to the elimination rule of the existential quantifier. If we have an witness for $Id_A(a, b)$, and if from a computational path between a and b we can construct a term of type C , then we can eliminate the identity type, obtaining a term of type C .

4 A Term Rewriting System for Paths

As we have just showed, a computational path establishes when two terms of the same type are equal. From the theory of computational paths, an interesting case arises. Suppose we have a path s that establishes that $a =_s b : A$ and a path t that establishes that $a =_t b : A$. Consider that s and t are formed by distinct compositions of rewrites. Is it possible to conclude that there are cases that s and t should be considered equivalent? The answer is *yes*. Consider the following example:

Example 4.1. Consider the path $a =_t b : A$. By the symmetric property, we obtain $b =_{\sigma(t)} a : A$. What if we apply the property again on the path $\sigma(t)$? We would obtain a path $a =_{\sigma(\sigma(t))} b : A$. Since we applied symmetry twice in succession, we obtained a path that is equivalent to the initial path t . For that reason, we conclude the act of applying symmetry twice in succession is a redundancy. We say that the path $\sigma(\sigma(t))$ can be reduced to the path t .

As one could see in the aforementioned example, different paths should be considered equal if one is just a redundant form of the other. The example that we have just seen is just a straightforward and simple case. Since the equality theory has a total of 7 axioms, the possibility of combinations that could generate redundancies are high. Fortunately, all possible redundancies were thoroughly mapped by de Oliveira (1995). In this work, a system that establishes all redundancies and creates rules that solve them was proposed. This system, known as $LND_{EQ} - TRS$, maps a total of 39 rules that solve redundancies. For each rule, there is a proof tree that constructs it. In the case of **example 4.1**, we have the following (de Queiroz & de Oliveira, 2013):

$$\frac{\frac{x =_t y : A}{y =_{\sigma(t)} x : A}}{x =_{\sigma(\sigma(t))} y : A} \triangleright_{ss} x =_t y : A$$

It is important to notice that we assign a label to every rule. In the previous case, we assigned the label ss .

Definition 4.2. An *rw*-rule is any of the rules defined in $LND_{EQ} - TRS$.

Definition 4.3. Let s and t be computational paths. We say that $s \triangleright_{1rw} t$ (read as: s *rw*-contracts to t) iff we can obtain t from s by an application of only one *rw*-rule. If s can be reduced to t by finite number of *rw*-contractions, then we say that $s \triangleright_{rw} t$ (read as s *rw*-reduces to t).

Definition 4.4. Let s and t be computational paths. We say that $s =_{rw} t$ (read as: s is *rw*-equal to t) iff t can be obtained from s by a finite (perhaps empty) series of *rw*-contractions and reversed *rw*-contractions. In other words, $s =_{rw} t$ iff there exists a sequence R_0, \dots, R_n , with $n \geq 0$, such that

$$\begin{aligned} (\forall i \leq n-1) (R_i \triangleright_{1rw} R_{i+1} \text{ or } R_{i+1} \triangleright_{1rw} R_i) \\ R_0 \equiv s, \quad R_n \equiv t \end{aligned}$$

Proposition 4.5. *is transitive, symmetric and reflexive.*

Proof. Comes directly from the fact that *rw*-equality is the transitive, reflexive and symmetric closure of *rw*. \square

We'd like to mention that $LND_{EQ} - TRS$ is terminating and confluent. The proof of this affirmation can be found in de Oliveira (1995); de Queiroz & de Oliveira (1994); de Oliveira & de Queiroz (1999); de Queiroz et al. (2011).

5 Homotopy Type Theory

In this section we develop the main objective of this work. We want to show that some of the foundational definitions, propositions and theorems of homotopy type theory still hold in our path-based approach. In other words, we use our approach to construct the building blocks of more complex results.

One important fact to notice is that every proof that does not involve the identity type is valid in the path-based approach. This is obvious, since the only difference between the traditional approach and ours is the formulation of the identity type. If a proof uses it, we need to reformulate this proof using our path-based approach, instead of using the induction principle of the traditional one. Thus, every part of a proof that is not directly or indirectly related to identity type is still valid in our approach.

In a path-based proof, we are going to use the formulation proposed in **section 3**. We also are going to use the reduction rules of $LND_{EQ} - TRS$. Every time we use a new rule, i.e., a reduction rule that was not previously defined, we are going to show its proof tree, as we did in the previous section for rule ss . In the process of developing the theory of this section, we noticed that $LND_{EQ} - TRS$, as proposed in de Oliveira (1995), is still incomplete. We state this based on the fact that we found new reduction rules that are not part of the original $LND_{EQ} - TRS$. That way, we added these new rules to the system, expanding it. One can check a full list of $LND_{EQ} - TRS$ rules in *appendix A*, i.e., a list containing the original 39 rules plus the ones newly found as a result of this work.

5.1 Groupoid Laws

One essential result of homotopy type theory is the fact that a type can be seen as a groupoid. This was first noticed in Hofmann & Streicher (1994). As shown by Hofmann & Streicher (1994) and further developed by Hofmann & Streicher (1998), the direct approach is to show that a set of laws known as groupoid laws holds up to propositional equality. Nevertheless, before we show that these rules hold for our path-based approach, we start showing that our identity type is, in fact, reflexive, symmetric and transitive.

Lemma 5.1. *The type $\Pi_{(a:A)} Id_A(a, a)$ is inhabited.*

Proof. We construct an witness for the desired type:

$$\frac{\frac{\frac{[a : A]}{a =_\rho a : A} Id - I_1}{\rho(a, a) : Id_A(a, a)} \Pi - I}{\lambda a. \rho(a, a) : \Pi_{(a:A)} Id_A(a, a)}$$

□

Lemma 5.2. *The type $\Pi_{(a:A)} \Pi_{(b:A)} (Id_A(a, b) \rightarrow Id_A(b, a))$ is inhabited.*

Proof. Similar to the previous lemma, we construct an witness:

$$\begin{array}{c}
\frac{[a =_t b : A]}{b =_{\sigma(t)} a : A} \\
\frac{[a : A] \quad [b : A] \quad \frac{(\sigma(t))(b, a) : Id_A(b, a)}{Id - I}}{[p(a, b) : Id_A(a, b)] \quad Id - E} \\
\frac{REWR(p(a, b), \acute{t}(\sigma(t))(b, a)) : Id_A(b, a)}{\Pi - I} \\
\frac{\lambda p. REWR(p(a, b), \acute{t}(\sigma(t))(b, a)) : Id_A(a, b) \rightarrow Id_A(b, a)}{\Pi - I} \\
\frac{\lambda b. \lambda p. REWR(p(a, b), \acute{t}(\sigma(t))(b, a)) : \Pi_{(b:A)}(Id_A(a, b) \rightarrow Id_A(b, a))}{\Pi - I} \\
\lambda a. \lambda b. \lambda p. REWR(p(a, b), \acute{t}(\sigma(t))(b, a)) : \Pi_{(a:A)} \Pi_{(b:A)}(Id_A(a, b) \rightarrow Id_A(b, a))
\end{array}$$

□

Lemma 5.3. *The type $\Pi_{(a:A)} \Pi_{(b:A)} \Pi_{(c:A)} (Id_A(a, b) \rightarrow Id_A(b, c) \rightarrow Id_A(a, c))$ is inhabited.*

Proof. We construct the following witness:

$$\begin{array}{c}
\frac{[a : A] \quad [b : A] \quad \frac{[c : A] \quad \frac{[a =_t b : A] \quad [b =_u c : A]}{a =_{\tau(t, u)} c : A} Id - I}{(\tau(t, u))(a, c) : Id_A(a, c)} Id - E \\
\frac{[w(a, b) : Id_A(a, b)] \quad \frac{REWR(s(b, c), \acute{u}(\tau(t, u))(a, c)) : Id_A(a, c)}{Id - E}}{REWR(w(a, b), \acute{t}REWR(s(b, c), \acute{u}(\tau(t, u))(a, c))) : Id_A(a, c)} \Pi - I \\
\frac{\lambda s. REWR(w(a, b), \acute{t}REWR(s(b, c), \acute{u}(\tau(t, u))(a, c))) : Id_A(b, c) \rightarrow Id_A(a, c)}{\Pi - I} \\
\frac{\lambda w. \lambda s. REWR(w(a, b), \acute{t}REWR(s(b, c), \acute{u}(\tau(t, u))(a, c))) : Id_A(a, b) \rightarrow Id_A(b, c) \rightarrow Id_A(a, c)}{\Pi - I} \\
\frac{\lambda b. \lambda w. \lambda s. REWR(w(a, b), \acute{t}REWR(s(b, c), \acute{u}(\tau(t, u))(a, c))) : \Pi_{(c:A)}(Id_A(a, b) \rightarrow Id_A(b, c) \rightarrow Id_A(a, c))}{\Pi - I} \\
\lambda a. \lambda b. \lambda c. \lambda w. \lambda s. REWR(w(a, b), \acute{t}REWR(s(b, c), \acute{u}(\tau(t, u))(a, c))) : \Pi_{(a:A)} \Pi_{(b:A)} \Pi_{(c:A)}(Id_A(a, b) \rightarrow Id_A(b, c) \rightarrow Id_A(a, c))
\end{array}$$

□

Lemma 5.1, **lemma 5.2** and **lemma 5.3** correspond respectively to the reflexivity, symmetry and transitivity of the identity type. From now on, the reflexivity will be represented by ρ , symmetry by σ and transitivity by τ .

Lemma 5.4. *Given a type A , $x, y, z, w : A$ and $p : Id_A(x, y)$ and $q : Id_A(y, z)$ and $r : Id_A(z, w)$, the following types are inhabited:*

1. $\Pi_{(x,y:A)} \Pi_{(p:Id_A(x,y))} Id_{Id_A(x,y)}(p, \rho_y \circ p)$ and $\Pi_{(x,y:A)} \Pi_{(p:Id_A(x,y))} Id_{Id_A(x,y)}(p, p \circ \rho_x)$.
2. $\Pi_{(x,y:A)} \Pi_{(p:Id_A(x,y))} Id_{Id_A(x,y)}(\sigma(p) \circ p, \rho_x)$ and $\Pi_{(x,y:A)} \Pi_{(p:Id_A(x,y))} Id_{Id_A(x,y)}(p \circ \sigma(p), \rho_y)$
3. $\Pi_{(x,y:A)} \Pi_{(p:Id_A(x,y))} Id_{Id_A(x,y)}(\sigma(\sigma(p)), p)$
4. $\Pi_{(x,y,z,w:A)} \Pi_{(p:Id_A(x,y))} \Pi_{(q:Id_A(y,z))} \Pi_{(r:Id_A(z,w))} Id_{Id_A(x,w)}(r \circ (q \circ p), (r \circ q) \circ p)$

Proof. The proof of each statement follows from the same idea. We just need to look for suitable reduction rules already present in the original $LND_{EQ} - TRS$.

1. The first thing to notice is that a composition in our path-based approach corresponds to a transitive operation, i.e., $(p \circ \rho_x)$ can be written as $\tau(\rho_x, p)$. Follows from rules number **5** and **6**. These are as follows:

$$\frac{x =_r y : A \quad y =_\rho y : A}{x =_{\tau(r, \rho)} y : A} \triangleright_{trr} \quad x =_r y : A$$

$$\frac{x =_{\rho} x : A \quad x =_r y : A}{x =_{\tau(\rho, r)} y : A} \triangleright_{tlr} \quad x =_r y : A$$

Thus, we have:

$$\frac{\frac{\tau(p, \rho_y) =_{trr} p : Id_A(x, y)}{(trr)(\tau(p, \rho_y), p) : Id_{Id_A(x, y)}(p, \rho_y \circ p)}}{\lambda x. \lambda y. \lambda p. (trr)(\tau(p, \rho_y), p) : \Pi_{(x, y: A)} \Pi_{(p: Id_A(x, y))} Id_{Id_A(x, y)}(p, \rho_y \circ p)}$$

$$\frac{\frac{\tau(\rho_x, p) =_{tlr} p : Id_A(x, y)}{(tlr)(\tau(\rho_x, p), p) : Id_{Id_A(x, y)}(p, p \circ \rho_x)}}{\lambda x. \lambda y. \lambda p. (tlr)(\tau(\rho_x, p), p) : \Pi_{(x, y: A)} \Pi_{(p: Id_A(x, y))} Id_{Id_A(x, y)}(p, p \circ \rho_x)}$$

2. We use rules **3** and **4**:

$$\frac{x =_r y : A \quad y =_{\sigma(r)} x : A}{x =_{\tau(r, \sigma(r))} x : A} \triangleright_{tr} \quad x =_{\rho} x : A$$

$$\frac{y =_{\sigma(r)} x : A \quad x =_r y : A}{y =_{\tau(\sigma(r), r)} y : A} \triangleright_{tsr} \quad y =_{\rho} y : A$$

Thus:

$$\frac{\frac{\tau(p, \sigma(p)) =_{tr} \rho_x : Id_A(x, y)}{(tr)(\tau(p, \sigma(p)), \rho_x) : Id_{Id_A(x, y)}(\sigma(p) \circ p, \rho_x)}}{\lambda x. \lambda y. \lambda p. (tr)(\tau(p, \sigma(p)), \rho_x) : \Pi_{(x, y: A)} \Pi_{(p: Id_A(x, y))} Id_{Id_A(x, y)}(\sigma(p) \circ p, \rho_x)}$$

$$\frac{\frac{\tau(\sigma(p), p) =_{tsr} \rho_y : Id_A(x, y)}{(tsr)(\tau(\sigma(p), p), \rho_y) : Id_{Id_A(x, y)}(p \circ \sigma(p), \rho_y)}}{\lambda x. \lambda y. \lambda p. (tsr)(\tau(\sigma(p), p), \rho_y) : \Pi_{(x, y: A)} \Pi_{(p: Id_A(x, y))} Id_{Id_A(x, y)}(p \circ \sigma(p), \rho_y)}$$

3. We use rule **2**:

$$\frac{\frac{x =_r y : A}{y =_{\sigma(r)} x : A}}{x =_{\sigma(\sigma(r))} y : A} \triangleright_{ss} \quad x =_r y : A$$

Thus:

$$\frac{\frac{\sigma(\sigma(p)) =_{ss} p : Id_A(x, y)}{(ss)(\sigma(\sigma(p)), p) : Id_{Id_A(x, y)}(\sigma(\sigma(p)), p)}}{\lambda x. \lambda y. \lambda p. (ss)(\sigma(\sigma(p)), p) : \Pi_{(x, y: A)} \Pi_{(p: Id_A(x, y))} Id_{Id_A(x, y)}(\sigma(\sigma(p)), p)}$$

4. We use rule **37**:

$$\begin{array}{c}
\frac{x =_t y : A \quad y =_r w : A}{x =_{\tau(t,r)} w : A} \quad w =_s z : A \\
\hline
x =_{\tau(\tau(t,r),s)} z : A \\
\\
\triangleright_{tt} \frac{x =_t y : A \quad \frac{y =_r w : A \quad w =_s z : A}{y =_{\tau(r,s)} z : A}}{x =_{\tau(t,\tau(r,s))} z : A}
\end{array}$$

Thus:

$$\frac{\frac{\tau(\tau(p,q),r) =_{tt} \tau(p,\tau(q,r)) : Id_A(x,w)}{(tt)(\tau(\tau(p,q),r) =_{tt} \tau(p,\tau(q,r))) : Id_{Id_A(x,w)}(r \circ (q \circ p), (r \circ q) \circ p)}}{\lambda x. \lambda y. \lambda z. \lambda w. \lambda p. \lambda q. \lambda r. (ss)(\sigma(\sigma(p),p)) : \Pi_{(p:Id_A(x,y))} \Pi_{(q:Id_A(y,z))} \Pi_{(r:Id_A(z,w))} Id_{Id_A(x,w)}(r \circ (q \circ p), (r \circ q) \circ p)}$$

□

With the previous lemma, we showed that our path-based approach yields the groupoid structure of a type up to propositional equality.

5.2 Functoriality

We want to show that functions preserve equality (Univalent Foundations Program, 2013).

Lemma 5.5. *The type $\Pi_{x,y:A} \Pi_{f:A \rightarrow B} (Id_A(x,y) \rightarrow Id_B(f(x),f(y)))$ is inhabited.*

Proof. It is a straightforward construction:

$$\frac{\frac{\frac{[x =_s y : A] \quad [f : A \rightarrow B]}{f(x) =_{\mu_f(s)} f(y) : B}}{\mu_f(s)(f(x),f(y)) : Id_B(f(x),f(y))} \quad [p : Id_A(x,y)]}{\frac{REWR(p, \lambda s. \mu_f(s)(f(x),f(y))) : Id_B(f(x),f(y))}{\lambda x. \lambda y. \lambda f. \lambda p. REWR(p, \lambda s. \mu_f(s)(f(x),f(y))) : \Pi_{x,y:A} \Pi_{f:A \rightarrow B} (Id_A(x,y) \rightarrow Id_B(f(x),f(y)))}}$$

□

Lemma 5.6. *Given functions $f : A \rightarrow B$ and $g : B \rightarrow C$ and paths $p : x =_A y$ and $q : y =_B z$, we have:*

1. $\mu_f(\tau(p,q)) = \tau(\mu_f(p), \mu_f(q))$
2. $\mu_f(\sigma(p)) = \sigma(\mu_f(p))$
3. $\mu_g(\mu_f(p)) = \mu_{g \circ f}(p)$
4. $\mu_{Id_A}(p) = p$

Proof. 1. For the first time, we needed to add a new rule to the original 39 rules of $LND_{EQ} - TRS$. We introduce rule **40**:

$$\frac{\frac{x =_p y : A \quad [f : A \rightarrow B]}{f(x) =_{\mu_f(p)} f(y) : B} \quad \frac{y =_q z : A \quad [f : A \rightarrow B]}{f(y) =_{\mu_f(q)} f(z) : B}}{f(x) =_{\tau(\mu_f(p), \mu_f(q))} f(z) : B}$$

$$\triangleright_{tf} \frac{x =_p y : A \quad y =_q z : A}{x =_{\tau(p,q)} z : A} \quad \frac{f : A \rightarrow B}{f(x) =_{\mu_f(\tau(p,q))} f(z) : B}$$

Thus, we have $\mu_f(\tau(p, q)) =_{\sigma(tf)} \tau(\mu_f(p), \mu_f(q))$

2. This one follows from rule **30**:

$$\frac{x =_p y : A \quad [f : A \rightarrow B]}{f(x) =_{\mu_f(p)} f(y) : B} \quad \frac{f(x) =_{\mu_f(p)} f(y) : B}{f(y) =_{\sigma(\mu_f(p))} f(x) : B}$$

$$\triangleright_{sm} \frac{x =_p y : A}{y =_{\sigma(p)} x : A} \quad \frac{[f : A \rightarrow B]}{f(y) =_{\mu_f(\sigma(p))} f(x) : B}$$

We have $\mu_f(\sigma(p)) =_{\sigma(sm)} \sigma(\mu_f(p))$

3. We introduce rule **41**:

$$\frac{x =_p y : A \quad [f : A \rightarrow B]}{f(x) =_{\mu_f(p)} f(y) : B} \quad \frac{[g : B \rightarrow C]}{g(f(x)) =_{\mu_g(\mu_f(p))} g(f(y)) : C}$$

$$\triangleright_{cf} \frac{x =_p y : A \quad \frac{[x : A] \quad [f : A \rightarrow B]}{f(x) : B} \quad \frac{[g : B \rightarrow C]}{g(f(x)) : C}}{\lambda x. g(f(x)) \equiv (g \circ f) : A \rightarrow C} \quad \frac{x =_p y : A}{g(f(x)) =_{\mu_{g \circ f}(p)} g(f(y)) : C}$$

Then, $\mu_g(\mu_f(p)) =_{cf} \mu_{g \circ f}(p)$

4. We use rule **42**:

$$\frac{x =_p y : A \quad [Id_A : A \rightarrow A]}{Id_A(x) = \mu_{Id_A}(p) Id_A(y) : A} \quad \triangleright_{ci} \quad x =_p y : A$$

Follows that $\mu_{Id_A}(p) =_{ci} p$

□

5.3 Transport

As stated in de Queiroz & de Oliveira (2014), substitution can take place when no quantifier is involved. In this sense, there is a 'quantifier-less' notion of substitution. In type theory, this 'quantifier-less' substitution is given by a operation known as transport (Univalent Foundations Program, 2013). In our path-based approach, we formulate a new inference rule of 'quantifier-less' substitution (de Queiroz & de Oliveira, 2014):

$$\frac{x =_p y : A \quad f(x) : P(x)}{p(x, y) \circ f(x) : P(y)}$$

We use this transport operation to solve one essential issue of our path-based approach. We know that given a path $x =_p y : A$ and function $f : A \rightarrow B$, the application of axiom μ yields the path $f(x) =_{\mu_f(p)} f(y) : B$. The problem arises when we try to apply the same axiom for a dependent function $f : \Pi_{(x:A)} P(x)$. In that case, we want $f(x) = f(y)$, but we cant guarantee that the type of $f(x) : P(x)$ is the same as $f(y) : P(y)$. The solution is to apply the transport operation and thus, we can guarantee that the types are the same:

$$\frac{x =_p y : A \quad f : \Pi_{(x:A)} P(x)}{p(x, y) \circ f(x) =_{\mu_f(p)} f(y) : P(y)}$$

Lemma 5.7. (Leibniz's Law) *The type $\Pi_{(x,y:A)} (Id_A(x, y) \rightarrow P(x) \rightarrow P(y))$ is inhabited.*

Proof. We construct the following tree:

$$\frac{\frac{\frac{[x =_p y : A] \quad [f(x) : P(x)]}{p(x, y) \circ f(x) : P(y)}}{\lambda f(x). p(x, y) \circ f(x) : P(x) \rightarrow P(y)} \quad [z : Id_A(x, y)]}{REWR(z, \lambda p. \lambda f(x). p(x, y) \circ f(x)) : P(x) \rightarrow P(y)} \quad \frac{}{\lambda x. \lambda y. \lambda z. REWR(z, \lambda p. \lambda f(x). p(x, y) \circ f(x)) : \Pi_{(x,y:A)} (Id_A(x, y) \rightarrow P(x) \rightarrow P(y))}$$

□

The function $\lambda f(x). p(x, y) \circ f(x) : P(x) \rightarrow P(y)$ is usually written as $transport^p(p, -)$ and $transport^p(p, f(x)) : P(y)$ is usually written as $p_*(f(x))$.

Lemma 5.8. *Given $P(x) \equiv B$, $x =_p y : A$ and $b : B$, there is a path $transport^P(p, b) = b$.*

Proof. The first to notice is the fact that in our formulation of transport, we always need a functional expression $f(x)$, and in this case we have only a constant term b . To address this problem, we consider a function $f = \lambda. b$ and then, we transport over $f(x) \equiv b$:

$$transport^P(p, f(x) \equiv b) =_{\mu(p)} (f(y) \equiv b).$$

Thus, $transport^P(p, b) =_{\mu(p)} b$. We sometimes call this path $transportconst_p^B(b)$.

□

Lemma 5.9. *Given $f : A \rightarrow B$ and $x =_p y : A$, we have*

$$\mu(p)(p * (f(x)), f(y)) = \tau(\text{transportconst}_p^B, \mu_f(p))(p * (f(x)), f(y))$$

Proof. The first thing to notice is that in this case, $\text{transportconst}_p^B$ is the path $\mu(p)(p * (f(x)), f(x))$ by lemma 5.8. As we did to the rules of $LND_{EQ} - TRS$, we establishes this equality by getting to the same conclusion from the same premises by two different trees:

In the first tree, we consider $f(x) \equiv b : B$ and transport over $b : B$:

$$\frac{\frac{x =_p y : A \quad f(x) \equiv b : B}{p(x, y) \circ (f(x) \equiv b) : B} \quad \frac{x =_p y : A \quad f : A \rightarrow B}{f(x) =_{\mu_f(p)} f(y) : B}}{\frac{p_*(f(x)) =_{\mu_f(p)} b \equiv f(x)}{p * (f(x)) =_{\tau(\mu_f(p), \mu_f(p))} f(y) : B}}$$

In the second one, we consider $f(x)$ as an usual functional expression and thus, we transport the usual way:

$$\frac{\frac{x =_p y : A \quad f(x) : B}{p(x, y) \circ f(x) : B}}{p_*(f(x)) =_{\mu_f(p)} f(y) : B}$$

□

Lemma 5.10. Given $x =_p y : A$ and $q : y =_A z : A$, $f(x) : P(x)$, we have

$$q_*(p_*(f(x))) = (p \circ q)_*(f(x))$$

Proof. We develop both sides of the equation and wind up with the same result:

$$\begin{aligned} q_*(p_*(f(x))) &=_{\mu(p)} q_*(f(y)) =_{\mu(q)} f(z) \\ (p \circ q)_*(f(x)) &=_{\mu(p \circ q)} f(z) \end{aligned}$$

□

Lemma 5.11. Given $f : A \rightarrow B$, $x =_p y : A$ and $u : P(f(x))$, we have:

$$\text{transport}^{P \circ f}(p, u) = \text{transport}^P(\mu_f(p), u)$$

Proof. This lemma hinges on the fact that there is two possible interpretations of u that stems from the fact that $(g \circ f)(x) \equiv g(f(x))$. Thus, we can see u as functional expression g on $f(x)$ or an expression $g \circ f$ on x :

$$\frac{\frac{x =_p y : A \quad u \equiv (g \circ f)(x) : (P \circ f)(x)}{p(x, y) \circ (g \circ f)(x) : (P \circ f)(y)} \quad \frac{\frac{x =_p y : A}{f(x) =_{u_f(p)} f(y) : B} \quad u \equiv g(f(x)) : P(f(x))}{\mu_f(p)(f(x), f(y)) \circ g(f(x)) : P(f(y))}}{\frac{p(x, y) \circ (g \circ f)(x) =_{\mu(p)} (g \circ f)(y) : (P \circ f)(y)}{p(x, y) \circ (g \circ f)(x) =_{\mu(p)} g(f(y)) : P(f(y))}} \quad \frac{\mu_f(p)(f(x), f(y)) \circ g(f(x)) =_{\mu(p)} g(f(y)) : P(f(y))}{g(f(y)) =_{\sigma(\mu(p))} \mu_f(p)(f(x), f(y)) \circ g(f(x)) : P(f(y))}}{\frac{p(x, y) \circ (g \circ f)(x) =_{\tau(\mu(p), \sigma(\mu(p)))} \mu_f(p)(f(x), f(y)) \circ g(f(x)) : P(f(y))}{\text{transport}^{P \circ f}(p, u) =_{\tau(\mu(p), \sigma(\mu(p)))} \text{transport}^P(\mu_f(p), u)}}$$

□

Lemma 5.12. Given $f : \Pi(x : A)P(x) \rightarrow Q(x)$, $x =_p y : A$ and $u(x) : P(x)$, we have:

$$\text{transport}^Q(p, f(u(x))) = f(\text{transport}^P(p, u(x)))$$

Proof. We proceed the usual way, constructing a derivation tree that establishes the equality:

$$\frac{\frac{x =_p y : A \quad f(u(x)) : Q(x)}{p(x, y) \circ f(u(x)) : Q(y)} \quad \frac{\frac{\frac{x =_p y : A \quad u(x) : P(x)}{p(x, y) \circ u(x) : P(y)} \quad f : \Pi(x : A)P(x) \rightarrow Q(x)}{p(x, y) \circ u(x) =_{\mu(p)} u(y) : P(y)} \quad \frac{f(p(x, y) \circ u(x)) =_{\mu_f(\mu(p))} f(u(y)) : Q(y)}{f(u(y)) =_{\sigma(\mu_f(\mu(p)))} f(p(x, y) \circ u(x)) : Q(y)}}{\frac{p(x, y) \circ f(u(x)) =_{\mu(p)} f(u(y)) : Q(y)}{p(x, y) \circ f(u(x)) =_{\tau(\mu(p), \sigma(\mu_f(\mu(p)))} f(p(x, y) \circ u(x))}}{\text{transport}^Q(p, f(u(x))) =_{\tau(\mu(p), \sigma(\mu_f(\mu(p)))} f(\text{transport}^P(p, u(x)))}$$

□

5.4 Homotopies

In Homotopy Type Theory, a homotopy is defined as follows (Univalent Foundations Program, 2013):

Definition 5.13. Given $f, g : \Pi(x : A)P(x)$, a homotopy from f to g is a dependent function of type:

$$(f \sim g) \equiv \Pi(x : A)(f(x) = g(x))$$

In our path-based approach, we have a homotopy $f, g : \Pi(x : A)P(x)$ if for every $x : A$ we have a computational path between $f(x) = g(x)$. Thus, if we have a homotopy $H_{f,g} : f \sim g$, we derive the following rule:

$$\frac{H_{f,g} : f \sim g \quad f, g : \Pi(x : A)P(x) \quad x : A}{f(x) =_{H_{f,g}(x)} g(x) : P(x)}$$

And:

$$\frac{f, g : \Pi(x : A)P(x) \quad x : A \quad [f, g : \Pi(x : A)P(x), x : A] \quad f(x) =_p g(x)}{H_{f,g}^p : f \sim g}$$

Lemma 5.14. Given $f, g, h : A \rightarrow B$, the following types are inhabited:

1. $f \sim f$
2. $(f \sim g) \rightarrow (g \sim f)$
3. $(f \sim g) \rightarrow (g \sim h) \rightarrow (f \sim h)$

Proof. 1. We construct the following term:

$$\frac{f : A \rightarrow B \quad x : A \quad \frac{[x : A] \quad x =_\rho x}{[f : A \rightarrow B] \quad f(x) =_{\mu_f(\rho)} f(x) : B}}{H_{f,f}^{\mu_f(\rho)} : f \sim f}$$

2. We construct:

$$\frac{\frac{f, g : A \rightarrow B \quad x : A}{\frac{[H_{f,g} : f \sim g] \quad [f, g : A \rightarrow B] \quad [x : A]}{f(x) =_{H_{f,g}(x)} g(x) : B}}}{g(x) =_{\sigma(H_{f,g}(x))} f(x) : B}}{\frac{H_{g,f}^{\sigma(H_{f,g}(x))}} : g \sim f}}{\lambda H_{f,g}. H_{g,f}^{\sigma(H_{f,g}(x))} : (f \sim g) \rightarrow (g \sim f)}$$

3. We construct:

$$\frac{\frac{f, h : A \rightarrow B \quad x : A}{\frac{[H_{f,g} : f \sim g] \quad [f, g : A \rightarrow B] \quad [x : A]}{f(x) =_{H_{f,g}(x)} g(x) : B}} \quad \frac{[H_{g,h} : g \sim h] \quad [g, h : A \rightarrow B]}{g(x) =_{H_{g,h}(x)} h(x) : B}}{f(x) =_{\tau(H_{f,g}(x), H_{g,h}(x))} h(x) : B}}{\frac{H_{f,h}^{\tau(H_{f,g}(x), H_{g,h}(x))}} : f \sim h}}{\lambda H_{f,g}. \lambda H_{g,h}. H_{f,h}^{\tau(H_{f,g}(x), H_{g,h}(x))} : (f \sim g) \rightarrow (g \sim h) \rightarrow (f \sim h)}$$

□

Lemma 5.15. Given $H_{f,g} : f \sim g$ and functions $f, g : A \rightarrow B$ and a path $x =_p y : A$ we have:

$$\tau(H_{f,g}(x), \mu_g(p)) = \tau(\mu_f(p), H_{f,g}(y))$$

Proof. To establish this equality, we need to add a new rule to our $LND_{EQ} - TRS$. We introduce rule **43**:

$$\frac{\frac{H_{f,g} : f \sim g \quad x : A \quad f, g : A \rightarrow B}{f(x) =_{H_{f,g}(x)} g(x) : B} \quad \frac{x =_p y : A}{g(x) =_{\mu_g(p)} g(y) : B}}{f(x) =_{\tau(H_{f,g}(x), \mu_g(p))} g(y) : B}}{\triangleright_{hp}}{\frac{x =_p y : A}{f(x) =_{\mu_f(p)} f(y) : B} \quad \frac{H_{f,g} : f \sim g \quad x : A \quad f, g : A \rightarrow B}{f(y) =_{H_{f,g}(y)} g(y) : B}}{f(x) =_{\tau(\mu_f(p), H_{f,g}(y))} g(y) : B}}$$

And thus:

$$\tau(H_{f,g}(x), \mu_g(p)) =_{hp} \tau(\mu_f(p), H_{f,g}(y))$$

□

After this subsection, we start to study specific lemmas and theorems involving basic types of type theory. Nevertheless, several of those theorems are statements about the notion of **equivalence** (notation: \simeq). Before we define equivalence, we need the following definition (Univalent Foundations Program, 2013):

Definition 5.16. A **quasi-inverse** of a function $f : A \rightarrow B$ is a triple (g, α, β) such that g is a function $g : B \rightarrow A$ and α and β are homotopies such that $\alpha : f \circ g \sim Id_B$ and $\beta : g \circ f \sim Id_A$

A quasi-inverse of f is usually written as $qinv(f)$.

Definition 5.17. A function $f : A \rightarrow B$ is an equivalence if there is a quasi-inverse $qinv(f) : B \rightarrow A$.

5.5 Cartesian Product

We start proving some important lemmas and theorems for the Cartesian product type. As we did in previous subsections, we proceed using our path-based approach. Before we prove our first theorem, it is important to remember that given a term $x : A \times B$, we can extract two projections, $FST(x) : A$ and $SND(x) : B$. Thus, given a path $x =_p y : A \times B$, we extract paths $FST(x) = FST(y) : A$ and $SND(x) = SND(y) : B$.

Theorem 5.18. *The function $(x =_p y : A \times B) \rightarrow (FST(x) = FST(y) : A) \times (SND(x) = SND(y) : B)$ is an equivalence for any x and y .*

Proof. To show the equivalence, we need to show the following

1. From $x =_p y : A \times B$ we want to obtain $(FST(x) = FST(y) : A) \times (SND(x) = SND(y) : B)$ and from that, we want to go back to $x =_p y : A \times B$.
2. We want to do the inverse process. From $(FST(x) = FST(y) : A) \times (SND(x) = SND(y) : B)$ we want to obtain $x =_p y : A \times B$ and then go back to $(FST(x) = FST(y) : A) \times (SND(x) = SND(y) : B)$.

To show the first part, we need rule **21**:

$$\frac{\frac{x =_p y : A \times B}{FST(x) =_{\mu_1(p)} FST(y) : A} \quad \frac{x =_p y : A \times B}{SND(x) =_{\mu_2(p)} SND(y) : B}}{\langle FST(x), SND(x) \rangle =_{\epsilon(\mu_1(p), \mu_2(p))} \langle FST(y), SND(y) \rangle : A \times B} \triangleright_{mx} x =_p y : A \times B.$$

Thus, applying rule mx we showed the first part of our proof. For the second part, we need rules **14** and **15**:

$$\frac{\frac{\frac{x =_r x' : A \quad y =_s z : B}{\langle x, y \rangle =_{\epsilon_\wedge(r,s)} \langle x', z \rangle : A \times B}}{FST(\langle x, y \rangle) =_{\mu_1(\epsilon_\wedge(r,s))} FST(\langle x', z \rangle) : A}}{\triangleright_{mx2l} x =_r x' : A.}$$

And:

$$\frac{\frac{\frac{x =_r y : A \quad z =_s w : B}{\langle x, z \rangle =_{\epsilon_\wedge(r,s)} \langle y, w \rangle : A \times B}}{FST(\langle x, z \rangle) =_{\mu_2(\epsilon_\wedge(r,s))} FST(\langle y, w \rangle) : B}}{\triangleright_{mx2r} z =_s w : B.}$$

We also use the η -reduction for the Cartesian product:

$$\langle FST(x), SND(x) \rangle : A \times B \triangleright_\eta x : A \times B$$

We construct the following derivation tree:

$$\frac{\frac{\frac{\langle FST(x) =_s FST(y), SND(x) =_t SND(y) \rangle}{FST(x) =_s FST(y) : A} \quad \frac{\langle FST(x) =_s FST(y), SND(x) =_t SND(y) \rangle}{SND(x) =_t SND(y) : B}}{\frac{\langle FST(x), SND(x) \rangle =_{\epsilon_\wedge(s,t)} \langle FST(y), SND(y) \rangle : A \times B}{x =_{\epsilon(s,t)} y : A \times B} \triangleright_\eta}$$

From $x =_{\epsilon(s,t)} y : A \times B$, we have:

$$\frac{\frac{x =_{\epsilon(s,t)} y : A \times B}{FST(x) =_{\mu_1(\epsilon_\wedge(s,t))} FST(y) : A} \quad \frac{x =_{\epsilon(s,t)} y : A \times B}{SND(x) =_{\mu_2(\epsilon_\wedge(s,t))} SND(y) : B}}{\frac{\langle FST(x) =_{\mu_1(\epsilon_\wedge(s,t))} FST(y), SND(x) =_{\mu_2(\epsilon_\wedge(s,t))} SND(y) \rangle}{\langle FST(x) =_s FST(y), SND(x) =_t SND(y) \rangle} \triangleright_{mx2l, mx2r} \wedge - I}$$

Thus, we showed part 2 and concluded the proof of this theorem. \square

Theorem 5.19. Given type families $\Pi_{z:Z} A, \Pi_{z:Z} B$ and a type family defined by $(A \times B)(z) \equiv A(z) \times B(z)$, a path $z =_p w : Z$ and $f(z) : A(z) \times B(z)$, we have:

$$transport^{A \times B}(p, f(z)) = \langle transport^A(p, FST(f(z))), transport^B(p, SND(f(z))) \rangle : A(w) \times B(w)$$

Proof. We construct a derivation tree that establishes the equality:

$$\frac{\frac{\frac{z =_p w : Z \quad f(z) : A(z) \times B(z)}{p(z, w) \circ f(z) : A(w) \times B(w)}}{p(z, w) \circ f(z) =_{\mu(p)} f(w) : A(w) \times B(w)}}{p(z, w) \circ f(z) =_{\tau(\mu(p), \eta)} \langle FST(f(w)), SND(f(w)) \rangle : A(w) \times B(w)} \quad \frac{\frac{\frac{z =_p w : Z \quad FST(f(z)) : A(z)}{p(z, w) \circ FST(f(z)) : A(w)} \quad \frac{z =_p w : Z \quad SND(f(z)) : B(z)}{p(z, w) \circ SND(f(z)) : B(w)}}{\langle p(z, w) \circ FST(f(z)), p(z, w) \circ SND(f(z)) \rangle : A(w) \times B(w)}}{\langle p(z, w) \circ FST(f(z)), p(z, w) \circ SND(f(z)) \rangle =_{\mu(p)} \langle FST(f(w)), SND(f(w)) \rangle}}{\langle FST(f(w)), SND(f(w)) \rangle =_{\sigma(\mu(p))} \langle p(z, w) \circ FST(f(z)), p(z, w) \circ SND(f(z)) \rangle}}{\frac{p(z, w) \circ f(z) =_{\tau(\tau(\mu(p), \eta), \sigma(\mu(p)))} \langle p(z, w) \circ FST(f(z)), p(z, w) \circ SND(f(z)) \rangle : A(w) \times B(w)}{transport^{A \times B}(p, f(z)) =_{\tau(\tau(\mu(p), \eta), \sigma(\mu(p)))} \langle transport^A(p, FST(f(z))), transport^B(p, SND(f(z))) \rangle : A(w) \times B(w)} \square$$

Theorem 5.20. Given $x, y : A \times B$, $FST(x) =_p FST(y) : A$, $SND(x) =_q SND(y) : B$, functions $g : A \rightarrow A'$, $h : B \rightarrow B'$ and $f : A \times B \rightarrow A' \times B'$ defined by $f(x) \equiv \langle g(FST(x)), h(SND(x)) \rangle$, we have:

$$\mu_f(\epsilon_\wedge(p, q)) = \epsilon_\wedge(\mu_g(p), \mu_h(q))$$

Proof. We introduce rule **44**:

$$\frac{\frac{\frac{FST(x) =_p FST(y) : A \quad SND(x) =_q SND(y) : B}{\langle FST(x), SND(x) \rangle =_{\epsilon_\wedge(p, q)} \langle FST(y), SND(y) \rangle : A \times B}}{x =_{\epsilon_\wedge(p, q)} y : A \times B}}{f(x) =_{\mu_f(\epsilon_\wedge(p, q))} f(y) : A' \times B'} =_\eta \triangleright_{mxc}$$

$$\frac{\frac{\frac{FST(x) =_p FST(y) : A}{g(FST(x)) =_{\mu_g(p)} g(FST(y)) : A'} \quad \frac{SND(x) =_q SND(y) : B}{h(SND(x)) =_{\mu_h(q)} h(SND(y)) : B'}}{\langle g(FST(x)), h(SND(x)) \rangle =_{\epsilon_\wedge(\mu_g(p), \mu_h(q))} \langle g(FST(y)), h(SND(y)) \rangle : A' \times B'}}{f(x) =_{\epsilon_\wedge(\mu_g(p), \mu_h(q))} f(y) : A' \times B'}$$

And thus:

$$\mu_f(\epsilon_\wedge(p, q)) =_{mxc} \epsilon_\wedge(\mu_g(p), \mu_h(q))$$

\square

5.6 Unit Type

For the unit type 1, our objective is to show the following theorem:

Theorem 5.21. *Given $x, y : 1$, there is a path t such that $x =_t y$. Moreover, $t = \rho$.*

Proof. To show that there is such t , we need to use the induction for the unit type (Univalent Foundations Program, 2013):

$$* \triangleright_\eta x : 1$$

Therefore, given $x, y : 1$, we have:

$$\frac{x =_{\sigma(\eta)} * : 1 \quad * =_\eta y : 1}{x =_{\tau(\sigma(\eta), \eta)} y : 1}$$

Moreover, by rule 4, we have:

$$\tau(\sigma(\eta), \eta) =_{tsr} \rho.$$

Thus, $t \equiv \tau(\sigma(\eta), \eta)$ and $t =_{tsr} \rho$. □

5.7 Function Extensionality

In this subsection, we are interested in the property of function extensionality. In other words, we want to conclude that given any two functions f, g , if for any x we have that $f(x) = g(x)$, then $f = g$. That $f = g$ implies $f(x) = g(x)$ by rules of basic type theory is showed in the sequel. Nonetheless, basic type theory is insufficient to derive function extensionality (Univalent Foundations Program, 2013). Therefore, we need to add a new rule. First, let's prove the following lemma:

Lemma 5.22. *The following function exists:*

$$(f = g) \rightarrow \Pi_{x:A} (f(x) = g(x) : B(x))$$

Proof. The construction is straightforward:

$$\frac{\frac{[f =_s g] \quad [x : A]}{f(x) =_{\nu(s)} g(x) : B(x)}}{\lambda s. \lambda x. (f(x) =_{\nu(s)} g(x)) : (f = g) \rightarrow \Pi_{x:A} (f(x) = g(x) : B(x))}$$

□

Now, to add function extensionality to our system, we add the following inference rule:

$$\frac{\lambda x. (f(x) =_t g(x)) : \Pi_{x:A} B}{f =_{ext(t)} g} \text{ ext}$$

Adding this rule to our system, we add the necessary tool to prove the following theorem:

Lemma 5.23. $(f = g) \simeq \Pi_{x:A} (f(x) = g(x) : B(x))$

Proof. This theorem is the direct application of two new reduction rules. The first one is rule **45**:

$$\frac{\frac{f =_s g : \Pi_{x:A} B \quad x : A}{f(x) =_{\nu(s)} g(x) : B(x)} \quad \lambda x. (f(x) =_{\nu(s)} g(x)) : \Pi_{x:A} B}{f =_{ext(\nu(s))} g : \Pi_{x:A} B} \triangleright_{extr} f =_s g : \Pi_{x:A} B$$

Thus:

$$ext(\nu(s)) =_{extr} s$$

The other one is rule **46**:

$$\frac{\frac{\lambda x. (f(x) =_t g(x)) : \Pi_{x:A} B}{f =_{ext(t)} g : \Pi_{x:A} B} \quad [x : A]}{\frac{f(x) =_{\nu(ext(t))} g(x) : B(x)}{\lambda x. (f(x) =_{\nu(ext(t))} g(x)) : \Pi_{x:A} B}} \triangleright_{extl} \lambda x. (f(x) =_t g(x)) : \Pi_{x:A} B$$

Therefore, we have:

$$\nu(ext(t)) =_{extl} t$$

Those two derivations tree establish the equivalence. \square

Before we prove the next theorem, we need to revisit transport. Given a function $f : A(x) \rightarrow B(x)$, it is possible to transport along this function f , resulting in $p_*(f) : A(y) \rightarrow B(y)$. In our approach, one should think of $p_*(f)$ as a function that has transport of a term $a : A(x)$ as input, i.e., $p_*(a) : A(y)$. Thus, we define $p_*(f)$ pointwise:

$$p_*(f)(p_*(a)) \equiv p_*(f(a))$$

Lemma 5.24. *Given a path $x =_p y : X$ and functions $f : A(x) \rightarrow B(x)$ and $g : A(y) \rightarrow B(y)$, we have the following equivalence:*

$$(p_*(f) = g) \simeq \Pi_{a:A(x)} (p_*(f(a)) = g(p_*(a)))$$

Proof. We give two derivations tree, using the rules that we have established in the previous theorem:

$$\frac{\frac{\frac{p_*(f) =_p g \quad [a : A(x)]}{p_*(f)(p_*(a)) =_{\nu(p)} g(p_*(a)) : B(y)}}{\lambda a. (p_*(f)(p_*(a)) \equiv f(a)) =_{\nu(p)} g(p_*(a)) : \Pi_{a:A(x)} (p_*(f(a)) = g(p_*(a)))}}{\frac{p_*(f) =_{ext(\nu(p))} g}{p_*(f) =_p g} \triangleright_{extl}}$$

And:

$$\begin{array}{c}
\frac{\lambda a. (p_*(f(a)) =_t g(p_*(a)))}{\lambda a. (p_*(f)(p_*(a)) =_t g(p_*(a)))} \\
\frac{p_*(f) =_{ext(t)} g \quad [a : A(x)]}{p_*(f)(p_*(a)) =_{\nu(ext(t))} g(p_*(a))} \\
\frac{p_*(f(a)) =_{\nu(ext(t))} g(p_*(a))}{p_*(f(a)) =_t g(p_*(a))} \triangleright_{extr} \\
\lambda a. (p_*(f(a)) =_t g(p_*(a)))
\end{array}$$

□

5.8 Univalence Axiom

The first thing to notice is that in our approach the following lemma holds:

Lemma 5.25. *Given types A and B , the following function exists:*

$$idtoeqv : (A = B) \rightarrow (A \simeq B)$$

Proof. The idea of the proof is similar to the one showed in Univalent Foundations Program (2013). We define $idtoeqv$ to be $p_* : A \rightarrow B$. Thus, to end this proof, we just need to show that p_* is an equivalence.

Given a path p , we can form a path $\sigma(p)$ and thus, we have $(\sigma(p))_* : B \rightarrow A$. Now, we show that $(\sigma(p))_*$ is a quasi-inverse of p_* .

We need to check that:

1. $p_*((\sigma(p))_*(b)) = b$
2. $(\sigma(p))_*(p_*(a)) = a$

Both equations can be showed by an application of **lemma 5.10**:

1. $p_*((\sigma(p))_*(b)) = (\sigma(p) \circ p)_*(b) = \tau(p, \sigma(p))_*(b) =_{tr} \rho_*(b) =_{\mu(p)} b$.
2. $(\sigma(p))_*(p_*(a)) = (p \circ \sigma(p))_*(a) = \tau(\sigma(p), p)_*(a) =_{tsr} \rho_*(a) =_{\mu(p)} a$

□

As we did in the previous subsection in **lemma 5.22**, we showed that a function exists, but we did not show that it is an equivalence. In fact, basic type theory cannot conclude that $idtoeqv$ is an equivalence Univalent Foundations Program (2013). If we want this equivalence to be a property of our system, we must add a new axiom. This axiom is known as Voevodsky's univalence axiom Univalent Foundations Program (2013):

Axiom 5.26. *For any types A, B , $idtoeqv$ is an equivalence, i.e., we have:*

$$(A = B) \simeq (A \simeq B)$$

Lemma 5.27. *Given $x, y : A$, $u(x) : B(x)$ and path $x =_p y : A$, we have:*

$$transport^B(p, u(x)) = transport^{X \rightarrow X}(\mu_B(p), u(x)) = idtoeqv(\mu_B(p))(u(x))$$

Proof. We develop every term of the equation and show that they arrive at the same conclusion:

$$\begin{array}{c}
\frac{x =_p y : A \quad u(x) : B(x)}{p(x, y) \circ u(x) : B(y)} \\
\hline
p(x, y) \circ u(x) =_{\mu(p)} u(y) : B(y) \\
\\
\frac{B(x) =_{\mu_B(p)} B(y) \quad u(x) : B(x)}{\mu_B(p)(B(x), B(y)) \circ u(x) : B(y)} \\
\hline
\mu_B(p)(B(x), B(y)) \circ u(x) =_{\mu(p)} u(y) : B(y)
\end{array}$$

Since $\text{idtoeqv} \equiv p_*$, we have that $\text{idtoeqv}(\mu_B(p))(u(x))$ is the same as $p_*(\mu_B(p))(u(x))$ that is the same as $\text{transport}^{X \rightarrow X}(\mu_B(p), u(x))$. \square

5.9 Identity Type

In this subsection, we investigate specific lemmas and theorems related to the identity type. We start with the following theorem:

Theorem 5.28. *if $f : A \rightarrow B$ is an equivalence, then for $x, y : A$ we have:*

$$\mu_f : (x = y : A) \rightarrow (f(x) = f(y) : B)$$

Proof. We will omit the specific details of this proof, since it is equal to the one of **theorem 2.11.1** presented in Univalent Foundations Program (2013). This is the case because this proof is independent of the usage of the induction principle of the identity type. The only difference is that at some steps we need to cancel inverse paths. In our approach, this is done by straightforward applications of **rules 3,4,5** and **6**. \square

Lemma 5.29. *For any $a : A$, with $x_1 =_p x_2$*

1. $\text{transport}^{x \rightarrow (a=x)}(p, q(x_1)) = \tau(q(x_1), p)$, *for $q(x_1) : a = x_1$*
2. $\text{transport}^{x \rightarrow (x=a)}(p, q(x_1)) = \tau(\sigma(p), q(x_1))$, *for $q(x_1) : x_1 = a$*
3. $\text{transport}^{x \rightarrow (x=x)}(p, q(x_1)) = \tau(\sigma(p), \tau(q(x_1), p))$ *for $q(x_1) : x_1 = x_1$*

Proof. 1. We start establishing the following reduction:

$$\frac{a =_{q(x_1)} x_1 \quad x_1 =_p x_2}{a =_{\tau(q(x_1), p)} x_2} \triangleright a =_{q(x_2)} x_2$$

Thus, we just need to show that $\text{transport}^{x \rightarrow (a=x)}(p, q(x_1))$ also reduces to $a =_{q(x_2)} x_2$:

$$\frac{x_1 =_p x_2 \quad q(x_1) : a = x_1}{p(x_1, x_2) \circ q(x_1) : a = x_2} =_{\mu(p)} (a =_{q(x_2)} x_2)$$

2. We use the same idea:

$$\frac{x_2 =_{\sigma(p)} x_1 \quad x_1 =_{q(x_1)} a}{x_2 =_{\tau(\sigma(p), q(x_1))} a} \triangleright x_2 =_{q(x_2)} a$$

$$\frac{x_1 =_p x_2 \quad q(x_1) : x_1 = a}{p(x_1, x_2) \circ q(x_1) : x_2 = a} =_{\mu(p)} (x_2 =_{q(x_2)} a)$$

3. Same as the previous cases:

$$\frac{\frac{x_2 =_{\sigma(p)} x_1 \quad x_1 =_{q(x_1)} x_1}{x_2 =_{\tau(\sigma(p), q(x_1))} x_1} \quad x_1 =_p x_2}{x_2 =_{\tau(\tau(\sigma(p), q(x_1)), p)} x_2} \triangleright x_2 =_{q(x_2)} x_2$$

$$\frac{x_1 =_p x_2 \quad q(x_1) : x_1 = x_1}{p(x_1, x_2) \circ q(x_1) : x_2 = x_2} =_{\mu(p)} (x_2 =_{q(x_2)} x_2)$$

□

Theorem 5.30. Given $f, g : A \rightarrow B$, with $a =_p a' : A$ and $f(a) =_{q(a)} g(a) : B$, we have:

$$\text{transport}^{x \rightarrow (f(x)=g(x)):B}(p, q) = \tau(\tau(\sigma(\mu f(p)), q(a)), \mu_g(p)) : f(a') = g(a')$$

Proof. This proof is analogous to the proof of the previous lemma:

$$\frac{\frac{a =_p a' : A}{f(a) =_{\mu_f(p)} f(a')}}{f(a') =_{\sigma(\mu_f(p))} f(a)} \quad \frac{f(a) =_{q(a)} g(a)}{f(a') =_{\tau(\sigma(\mu_f(p)), q(a))} g(a)} \quad \frac{a =_p a'}{g(a) =_{\mu_g(p)} g(a')} \triangleright f(a') =_{q(a')} g(a')$$

And:

$$\frac{a =_p a' \quad q(a) : f(a) = g(a)}{p(a, a') \circ q(a) : f(a') = g(a')} =_{\mu(p)} (f(a') =_{q(a')} g(a'))$$

□

Theorem 5.31. Given $f, g : \Pi_{(x:A)} B(x)$, with $a =_p a' : A$ and $f(a) =_{q(a)} g(a) : B(a)$, we have:

$$\text{transport}^{x \rightarrow (f(x)=g(x)):B(x)}(p, q) = \tau(\tau(\sigma(\text{apd}_f(p)), \mu_{\text{transport}^{B_p}}(q)), \text{apd}_g(p))$$

where $\text{apd}_f(p) \equiv (p(a, a') \circ f(a) =_{\mu(p)} f(a'))$ and $\text{apd}_g \equiv (p(a, a') \circ g(a) =_{\mu(p)} g(a'))$

Proof. Similar to previous theorem:

$$\frac{\frac{p(a, a') \circ f(a) =_{\mu(p)} f(a')}{f(a') =_{\sigma(\mu(p))} p(a, a') \circ f(a)} \quad \frac{f(a) =_{q(a)} g(a)}{p(a, a') \circ f(a) =_{\mu_{\text{transport}^{B_p}}(q(a))} p(a, a') \circ g(a)}}{f(a') =_{\tau(\sigma(\mu(p)), \mu_{\text{transport}^{B_p}}(q(a)))} p(a, a') \circ g(a)} \quad p(a, a') \circ g(a) =_{\mu(p)} g(a') \triangleright f(a') =_{q(a')} g(a')$$

And:

$$\frac{a =_p a' \quad q(a) : f(a) = g(a)}{p(a, a') \circ q(a) : f(a') = g(a')} \triangleright_{\mu(p)} f(a') =_{q(a')} g(a')$$

□

Theorem 5.32. Given $a =_p a' : A$, $a =_q a$ and $a' =_r a'$, we have:

$$(transport^{x \rightarrow (x=x)}(p, q) = r) \simeq (\tau(q, p) = \tau(p, r))$$

Proof. We use **lemma 5.29** to prove this theorem, together with **rules 3,4,5,6** and **37**. We also consider functions $f(x) \equiv \tau(p, x) : (a' = z) \rightarrow (a = z)$ and $f^{-1}(x) \equiv \tau(\sigma(p), x) : (a = z) \rightarrow (a' = z)$. We proceed the same way as we have done to prove previous equivalences. In other words, we show two derivations trees. They are as follows:

$$\frac{\frac{\frac{transport^{x \rightarrow (x=x)}(p, q) = r}{\tau(\sigma(p), \tau(q, p)) = r} \text{ lemma 5.29}}{\tau(p, \tau(\sigma(p), \tau(q, p))) = \tau(p, r)} \mu_f}{\tau(\tau(p, \sigma(p)), \tau(q, p)) = \tau(p, r)} \frac{\tau(\rho, \tau(q, p)) = \tau(p, r)}{\tau(q, p) = \tau(q, p)}$$

And:

$$\frac{\frac{\frac{\tau(q, p) = \tau(p, r)}{\tau(\sigma(p), \tau(q, p)) = \tau(\sigma(p), \tau(p, r))} \mu_{f^{-1}}}{\tau(\sigma(p), \tau(q, p)) = \tau(\tau(\sigma(p), p), r)} \frac{\tau(\sigma(p), \tau(q, p)) = \tau(\tau(\rho, r))}{\tau(\sigma(p), \tau(q, p)) = r} \text{ lemma 5.29}}{transport^{x \rightarrow (x=x)}(p, q) = r}$$

□

5.10 Coproduct

One essential thing to remember is that a product $A + B$ has a left injection $inl : A \rightarrow A + B$ and $inr : B \rightarrow A + B$. As described in Univalent Foundations Program (2013), it is expected that $A + B$ contains copies of A and B disjointly. In our path based approach, we achieve this by constructing every path $inl(a) = inl(b)$ and $inr(a) = inr(b)$ by applications of axiom μ on paths $a = b$. Thus we show that we get the following equivalences:

1. $(inl(a_1) = inl(a_2)) \simeq (a_1 = a_2)$
2. $(inr(b_1) = inr(b_2)) \simeq (b_1 = b_2)$
3. $(inl(a) = inr(b)) \simeq 0$

To prove this, we use the same idea as in Univalent Foundations Program (2013). We characterize the type:

$$(x \rightarrow (inl(a_0) = x)) : \Pi_{(x:A+B)}(inl(a_0 = x))$$

To do this, we define a type *code*:

$$x : A + B \vdash code(x) \text{ type}$$

Our main objective is to prove the equivalence $\Pi_{(x:A+B)}((inl(a_0) = x) \simeq code(x))$. Using the recursion principle of the coproduct, we can define *code* by two equations:

$$\begin{aligned} code(inl(a)) &\equiv (a_0 = a) \\ code(inr(b)) &\equiv 0 \end{aligned}$$

Theorem 5.33. *Given $x : A + B$, we have $inl(a_0 = x) \simeq code(x)$*

Proof. To show this equivalence, we use the same method as the one showed in Univalent Foundations Program (2013). The main idea is to define functions

$$\begin{aligned} encode &: \Pi_{(x:A+B)} \Pi_{(p:inl(a_0)=x)} code(x) \\ decode &: \Pi_{(x:A+B)} \Pi_{(c:code(x))} (inl(a_0) = x) \end{aligned}$$

such that *decode* acts as a quasi-inverse of *encode*.

We start defining *encode*:

$$encode(x, s) \equiv transport^{code}(s, \rho_{a_0})$$

We notice that $\rho_{a_0} : code(inl(a_0))$, since $code(inl(a_0)) \equiv (a_0 =_p a_0)$. We also notice that for *encode*, it is only possible for the argument x to be of the form $x \equiv inl(a)$, since the other possibility is $x \equiv inr(a)$, but that case is not possible, because we would have a function to $code(inr(b)) \equiv 0$.

For *decode*, when $x \equiv inl(a)$, we have that $code(x) \equiv a_0 =_c a$ and thus, we define *decode* as $(inl(a_0) =_{\mu(c)} inl(a))$. When $x \equiv inr(a)$, then $code(x) \equiv 0$ and thus, we define *decode* as having any value, given by the elimination of the type 0. Now, we can finally prove the equivalence.

Starting with *encode*, we have $x \equiv inl(a)$, $inl(a_0) =_s x$. Since $encode(x, s) \equiv transport^{code}(s, \rho_{a_0})$, we have:

$$\frac{\frac{inl(a_0) =_s inl(a) \quad \rho_{a_0} : code(inl(a_0))}{s(inl(a_0), inl(a)) \circ \rho_{a_0} : code(inl(a))} =_{\mu(s)} \rho_a : code(inl(a)) \equiv code(x)}$$

Now, we can go back to $inl(a_0) = inl(a)$ by an application of *decode*, since:

$$decode(\rho_a : code(x)) \equiv inl(a_0) =_{\mu_{inl}} inl(a)$$

And we conclude this part, since in our approach $inl(a_0) =_s inl(a)$ is constructed by applications of axiom μ .

Now, we start from *decode*. Let $c : code(x)$. If $x \equiv inl(a)$, then $c : a_0 = a$ and thus, $decode(c) \equiv inl(a_0) =_{\mu(c)} inl(a)$. Now, we apply *encode*. We have:

$$\begin{aligned} encode(x, \mu_c) &= transport^{code}(\mu_c, \rho_{a_0}) \\ &= transport^{a \rightarrow (a_0=a)}(c, \rho_{a_0}) && \textbf{(Lemma 5.11)} \\ &= \tau(\rho_{a_0}, c) && \textbf{(Lemma 5.29)} \\ &= c && \textbf{(Rule 6)} \end{aligned}$$

If $x \equiv inr(b)$, we have that $c : 0$ and thus, as stated in Univalent Foundations Program (2013), we can conclude anything we wish. □

5.11 Natural Numbers

In our approach, the path space of the naturals are characterized by a trivial path $0 =_\rho 0$, and that every other path is constructed by the application of axiom μ together with function succ . We show that this characterization is similar to the one constructed in Univalent Foundations Program (2013). To do this, we use code , encode and decode . For \mathbb{N} , we define code recursively (Univalent Foundations Program, 2013):

$$\begin{aligned} \text{code}(0, 0) &\equiv 1 \\ \text{code}(\text{succ}(m), 0) &\equiv 0 \\ \text{code}(0, \text{succ}(m)) &\equiv 0 \\ \text{code}(\text{succ}(m), \text{succ}(n)) &\equiv \text{code}(m, n) \end{aligned}$$

We also define a dependent function $r : \Pi_{(n:\mathbb{N})} \text{code}(m, n)$, with:

$$\begin{aligned} r(0) &\equiv * \\ r(\text{succ}(n)) &\equiv r(n) \end{aligned}$$

Before we proceed to the next theorem, we need to establish a new reduction rule. We introduce **rule 47**:

$$\frac{x =_{\rho_x} x : A \quad [f : \Pi_{(x:A)} B(x)]}{f(x) =_{\mu(\rho_x)} f(x) : B(x)} \triangleright_{\text{mxp}} \quad f(x) =_{\rho_{f(x)}} f(x)$$

Thus, $\mu_f(\rho_x) =_{\text{mxp}} \rho_{f(x)}$.

We also need the following lemma:

Lemma 5.34. *Given $m, n : \mathbb{N}$, if there is a path $m =_t n : \mathbb{N}$, then $t \triangleright \rho$.*

Proof. We prove by induction. The base is trivial, since $0 =_\rho 0$. Now, consider that there is a path $\text{succ}(m) =_{t'} \text{succ}(n)$. If t' is a trivial path, we end the proof, since $t' = \rho$. The other possibility is that t' is constructed by applications of axiom μ . In that case, there is a $m =_t n$ such that $t' \equiv \mu_{\text{succ}}(t)$. By the inductive hypothesis, $t \triangleright \rho$. Thus, $t' \equiv \mu_{\text{succ}}(t) \triangleright \mu_{\text{succ}}(\rho) =_{\text{mxp}} \rho_{\text{succ}}$. Thus, $t' \triangleright \rho$. \square

Theorem 5.35. *Given $m, n : \mathbb{N}$, we have $(m = n) \simeq \text{code}(m, n)$*

Proof. We need to define encode and decode and prove that they are pseudoinverses. We define $\text{encode} : \Pi_{(m,n:\mathbb{N})} (m = n) \rightarrow \text{code}(m, n)$ as:

$$\text{encode}(m, n, p) \equiv \text{transport}^{\text{code}(m, -)}(p, r(m))$$

We define $\text{decode} : \Pi_{(m,n:\mathbb{N})} \text{code}(m, n) \rightarrow (m = n)$ recursively:

$$\begin{aligned} \text{decode}(0, 0, c) &\equiv 0 =_\rho 0 \\ \text{decode}(\text{succ}(m), 0, c) &\equiv 0 \\ \text{decode}(0, \text{succ}(m), c) &\equiv 0 \\ \text{decode}(\text{succ}(m), \text{succ}(n), c) &\equiv \mu_{\text{succ}}(\text{decode}(m, n, c)) \end{aligned}$$

We now prove that if $m =_p n$, then $\text{decode}(\text{code}(m, n)) = \rho$. We prove by induction. The base is trivial, since $\text{decode}(0, 0, c) \equiv \rho$. Now, consider $\text{decode}(\text{succ}(m), \text{succ}(n), c)$. We have that $\text{decode}(\text{succ}(m), \text{succ}(n), c) \equiv \mu_{\text{succ}}(\text{decode}(m, n, c))$. By the inductive hypothesis, $\text{decode}(m, n, c) \equiv \rho$. Thus, we need to prove that $\mu_{\text{succ}} = \rho$. This last step is a straightforward application of **rule 47**. Therefore, $\mu_{\text{succ}} =_{\text{mxp}} \rho$. With this information, we can start the proof of the equivalence.

Given $m =_p n$, we have:

$$\text{encode}(m, n, p) \equiv \text{transport}^{\text{code}(m, -)}(p, r(m))$$

Thus:

$$\frac{m =_p n \quad r(m) : \text{code}(m, m)}{p(m, n) \circ r(m) : \text{code}(m, n)} =_{\mu(p)} (r(n) : \text{code}(m, n))$$

Now, we know that $\text{decode}(r(n) : \text{code}(m, n)) = \rho$ and, by **lemma 5.34**, $p = \rho$.

The proof starting from a $c : \text{code}(m, n)$ is equal to the one presented in Univalent Foundations Program (2013). We prove by induction. If m and n are 0, we have the trivial path $0 =_\rho 0$, thus $\text{decode}(0, 0, c) = \rho_0$, whereas $\text{encode}(0, 0, \rho_0) \equiv r(0) \equiv *$. We conclude this part recalling that every $x : 1$ is equal to $*$, since we have $x =_\sigma (\eta) * : 1$. In the case of $\text{decode}(\text{succ}(m), 0, c)$ or $\text{decode}(0, \text{succ}(n), c)$, $c : 0$. The only case left is for $\text{decode}(\text{succ}(m), \text{succ}(n), c)$. Similar to Univalent Foundations Program (2013), we prove by induction:

$$\begin{aligned} & \text{encode}(\text{succ}(m), \text{succ}(n), \text{decode}(\text{succ}(m), \text{succ}(n), c)) \\ &= \text{encode}(\text{succ}(m), \text{succ}(n), \mu_{\text{succ}}(\text{decode}(m, n, c))) \\ &= \text{transport}^{\text{code}(\text{succ}(m), -)}(\mu_{\text{succ}}(\text{decode}(m, n, c)), r(\text{succ}(m))) \\ &= \text{transport}^{\text{code}(\text{succ}(m), \text{succ}(-))}(\text{decode}(m, n, c), r(\text{succ}(m))) \\ &= \text{transport}^{\text{code}(m, -)}(\text{decode}(m, n, c), r(m)) \\ &= \text{encode}(m, n, \text{decode}(m, n, c)) \\ &= c \end{aligned}$$

□

6 Conclusion and Future Work

Inspired by a recent discovery that the propositional equality between terms can be interpreted as the type of homotopy paths, we have revisited the formulation of the intensional identity type, proposing a new approach based on an entity known as *computational path*. We have proposed that a computational path $a =_s b : A$ gives grounds to building a term $s(a, b)$ of the identity type, i.e., $s(a, b) : \text{Id}_A(a, b)$, and is formed by a composition of basic rewrites, each with their identifiers taken as constants. We have also developed our approach, showing how the path-based identity type can be rather straightforwardly used in deductions. In particular, we have shown the simplicity of our elimination rule, demonstrating that it is based on path constructions, which are built from applications of simple axioms of equality for type theory.

After establishing the foundations of our approach, we proceeded recalling the concept of term rewrite system, also known as $\text{LND}_{EQ} - \text{TRS}$. This system establish rules of reduction between paths, i.e., it effectively establishes an algebra of paths. Thus, we have used this algebra to establish important results of Homotopy Type Theory. We showed a total of 21 lemmas and 10 theorems. That way, we have showed that our approach yields the main building blocks of Homotopy Type Theory, on par with the classic approach. We have also improved the rewrite system, adding 8 new reduction rules. In view of these results, we have developed a valid alternative approach to the identity type and homotopy type theory, based on this algebra of paths. We also believe that we have opened the way, in future works, for possible expansions of this results, formulating and proving even more intricate concepts and theorems of homotopy type theory using computational paths.

Since our results tied in with the ones already established in the classic approach for the identity type, it is also possible to investigate in future works

the structure of our identity type. In other words, investigate the possibility of this path-approach inducing a ω -groupoid. We expect this result, since it has been proved by Lumsdaine (2009); van den Berg & Garner (2011) for the classical approach.

References

- de Oliveira, A. G. (1995), '*Proof transformations for labelled natural deduction via term rewriting*'. Master's thesis, Depto. de Informática, Universidade Federal de Pernambuco, Recife, Brazil, April 1995.
- de Oliveira, A. G. & de Queiroz, R. J. G. B. (1999), 'A normalization procedure for the equational fragment of labelled natural deduction', *Logic Journal of IGPL* **7**(2), 173–215.
- de Queiroz, R. J. G. B. & de Oliveira, A. G. (1994), Term rewriting systems with labelled deductive systems, in 'Proceedings of Brazilian Symposium on Artificial Intelligence (SBIA'94)', pp. 59–72.
- de Queiroz, R. J. G. B. & de Oliveira, A. G. (2013), 'Propositional equality, identity types, and direct computational paths'. <http://arxiv.org/abs/1107.1901>.
- de Queiroz, R. J. G. B. & de Oliveira, A. G. (2014), Natural deduction for equality: The missing entity, in 'Advances in Natural Deduction', Springer, pp. 63–91.
- de Queiroz, R. J. G. B. & Gabbay, D. M. (1994), Equality in labelled deductive systems and the functional interpretation of propositional equality, in 'Proceedings of the 9th Amsterdam Colloquium', ILLC/Department of Philosophy, University of Amsterdam, pp. 547–565.
- de Queiroz, R. J. G. B., de Oliveira, A. G. & Gabbay, D. M. (2011), *The Functional Interpretation of Logical Deduction*, World Scientific.
- Harper, R. (2012), 'Type theory foundations'. Type Theory Foundations, Lecture at Oregon Programming Languages Summer School, Eugene, Oregon.
- Hindley, J. R. & Seldin, J. P. (2008), *Lambda-calculus and combinators: an introduction*, Cambridge University Press.
- Hofmann, M. & Streicher, T. (1994), The groupoid model refutes uniqueness of identity proofs, in 'Logic in Computer Science, 1994. LICS'94. Proceedings., Symposium on', IEEE, pp. 208–212.
- Hofmann, M. & Streicher, T. (1998), The groupoid interpretation of type theory, in 'Twenty-five years of constructive type theory (Venice, 1995)', Vol. 36 of *Oxford Logic Guides*, Oxford Univ. Press, New York, pp. 83–111.
- Le Chenadec, P. (1989), 'On the logic of unification', *Journal of Symbolic computation* **8**(1), 141–199.
- Lumsdaine, P. L. (2009), Weak ω -categories from intensional type theory, in 'Typed lambda calculi and applications', Vol. 5608 of *LNCS*, Springer, pp. 172–187.
- Prawitz, D. (2009), Inference and knowledge, in 'The Logica Yearbook 2008', College Publications, London, pp. 175–192.

- Univalent Foundations Program, T. (2013), *Homotopy Type Theory: Univalent Foundations of Mathematics*, <https://homotopytypetheory.org/book>, Institute for Advanced Study.
- van den Berg, B. & Garner, R. (2011), ‘Types are weak ω -groupoids’, *Proceedings of the London Mathematical Society* **102**(2), 370–394.
- Voevodsky, V. (n.d.), ‘Univalent foundations and set theory’. Univalent Foundations and Set Theory, Lecture at IAS, Princeton, New Jersey, Mar 2014.

A Subterm Substitution

In Equational Logic, the subterm substitution is given by the following inference rule (de Queiroz & de Oliveira, 1994):

$$\frac{s = t}{s\theta = t\theta}$$

One problem is that such rule does not respect the subformula property. To deal with that, Le Chenadec (1989) proposes two inference rules:

$$\frac{M = N \quad C[N] = O}{C[M] = O} IL \quad \frac{M = C[N] \quad N = O}{M = C[O]} IR$$

where M, N and O are terms.

As proposed in de Queiroz & de Oliveira (2013), we can define similar rules using computational paths, as follows:

$$\frac{x =_r C[y] : A \quad y =_s u : A'}{x =_{\text{sub}_L(r,s)} C[u] : A} \quad \frac{x =_r w : A' \quad C[w] =_s u : A}{C[x] =_{\text{sub}_R(r,s)} u : A}$$

where C is the context in which the subterm detached by '[]' appears and A' could be a subdomain of A , equal to A or disjoint to A .

In the rule above, $C[u]$ should be understood as the result of replacing every occurrence of y by u in C .

B List of Rewrite Rules

We present all rewrite rules of $LND_{EQ} - TRS$. They are as follows (All have been taken from de Queiroz & de Oliveira (2013)):

1. $\sigma(\rho) \triangleright_{sr} \rho$
2. $\sigma(\sigma(r)) \triangleright_{ss} r$
3. $\tau(\mathcal{C}[r], \mathcal{C}[\sigma(r)]) \triangleright_{tr} \mathcal{C}[\rho]$
4. $\tau(\mathcal{C}[\sigma(r)], \mathcal{C}[r]) \triangleright_{tsr} \mathcal{C}[\rho]$
5. $\tau(\mathcal{C}[r], \mathcal{C}[\rho]) \triangleright_{trr} \mathcal{C}[r]$
6. $\tau(\mathcal{C}[\rho], \mathcal{C}[r]) \triangleright_{tlr} \mathcal{C}[r]$
7. $\text{sub}_L(\mathcal{C}[r], \mathcal{C}[\rho]) \triangleright_{slr} \mathcal{C}[r]$
8. $\text{sub}_R(\mathcal{C}[\rho], \mathcal{C}[r]) \triangleright_{srr} \mathcal{C}[r]$
9. $\text{sub}_L(\text{sub}_L(s, \mathcal{C}[r]), \mathcal{C}[\sigma(r)]) \triangleright_{sls} s$
10. $\text{sub}_L(\text{sub}_L(s, \mathcal{C}[\sigma(r)]), \mathcal{C}[r]) \triangleright_{slss} s$
11. $\text{sub}_R(\mathcal{C}[s], \text{sub}_R(\mathcal{C}[\sigma(s)], r)) \triangleright_{srs} r$
12. $\text{sub}_R(\mathcal{C}[\sigma(s)], \text{sub}_R(\mathcal{C}[s], r)) \triangleright_{srrr} r$
13. $\mu_1(\xi_1(r)) \triangleright_{mx2l1} r$
14. $\mu_1(\xi_\wedge(r, s)) \triangleright_{mx2l2} r$
15. $\mu_2(\xi_\wedge(r, s)) \triangleright_{mx2r1} s$
16. $\mu_2(\xi_2(s)) \triangleright_{mx2r2} s$
17. $\mu(\xi_1(r), s, u) \triangleright_{mx3l} s$
18. $\mu(\xi_2(r), s, u) \triangleright_{mx3r} u$
19. $\nu(\xi(r)) \triangleright_{mxl} r$
20. $\mu(\xi_2(r), s) \triangleright_{mxr} s$
21. $\xi(\mu_1(r), \mu_2(r)) \triangleright_{mx} r$
22. $\mu(t, \xi_1(r), \xi_2(s)) \triangleright_{mxx} t$
23. $\xi(\nu(r)) \triangleright_{xmr} r$
24. $\mu(s, \xi_2(r)) \triangleright_{mx1r} s$
25. $\sigma(\tau(r, s)) \triangleright_{stss} \tau(\sigma(s), \sigma(r))$
26. $\sigma(\text{sub}_L(r, s)) \triangleright_{ssbl} \text{sub}_R(\sigma(s), \sigma(r))$
27. $\sigma(\text{sub}_R(r, s)) \triangleright_{ssbr} \text{sub}_L(\sigma(s), \sigma(r))$
28. $\sigma(\xi(r)) \triangleright_{sx} \xi(\sigma(r))$
29. $\sigma(\xi(s, r)) \triangleright_{sxss} \xi(\sigma(s), \sigma(r))$
30. $\sigma(\mu(r)) \triangleright_{sm} \mu(\sigma(r))$
31. $\sigma(\mu(s, r)) \triangleright_{smss} \mu(\sigma(s), \sigma(r))$
32. $\sigma(\mu(r, u, v)) \triangleright_{smsss} \mu(\sigma(r), \sigma(u), \sigma(v))$
33. $\tau(r, \text{sub}_L(\rho, s)) \triangleright_{tsbl} \text{sub}_L(r, s)$
34. $\tau(r, \text{sub}_R(s, \rho)) \triangleright_{tsbrl} \text{sub}_L(r, s)$
35. $\tau(\text{sub}_L(r, s), t) \triangleright_{tsblr} \tau(r, \text{sub}_R(s, t))$
36. $\tau(\text{sub}_R(s, t), u) \triangleright_{tsbrrr} \text{sub}_R(s, \tau(t, u))$
37. $\tau(\tau(t, r), s) \triangleright_{tt} \tau(t, \tau(r, s))$
38. $\tau(\mathcal{C}[u], \tau(\mathcal{C}[\sigma(u)], v)) \triangleright_{tts} v$
39. $\tau(\mathcal{C}[\sigma(u)], \tau(\mathcal{C}[u], v)) \triangleright_{tst} u$
40. $\tau(\mu(r), \mu(s)) =_{tf} \mu(\tau(r, s))$
41. $\mu_g(\mu_f(p)) =_{cf} \mu_{g \circ f}(p)$
42. $\mu_{Id_A}(p) =_{ci} p$
43. $\tau(H_{f,g}(x), \mu_g(p)) =_{hp} \tau(\mu_f(p), H_{f,g}(y))$
44. $\mu_f(\epsilon_\wedge(p, q)) =_{mxc} \epsilon_\wedge(\mu_g(p), \mu_h(q))$
45. $\text{ext}(\nu(s)) =_{extr} s$
46. $\nu(\text{ext}(t)) =_{extl} t$

$$47. \mu_f(\rho_x) =_{\max} \rho_{f(x)} \cdot \cdot$$